UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/885,427	06/19/2001	Peter A.J. van der Made	81924.0002	4888

7590 01/04/2005

W SCOTT PETTY
KING & SPALDING
191 PEACHTREE STREET 45TH FLOOR
ATLANTA, GA 30303-1763

EXAMINER	
GUILL, RUSSELL L	
ART UNIT	**PAPER NUMBER**
2123	

DATE MAILED: 01/04/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/885,427	MADE, PETER A.J. VAN DER	
	Examiner	Art Unit	
	Russell L. Guill	2123	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *6/19/2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1 - 10* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1 - 10* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on *09/07/2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08). Paper No(s)/Mail Date 5. 2 pages
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1 – 10 have been examined. Claims 1 – 10 have been rejected.

Specification

2. The use of the trademarks WINDOWS, LINUX, VISUAL BASIC, PENTIUM, and PERL have been noted in this application. They should be capitalized wherever they appear and be accompanied by the generic terminology.

   Although the use of trademarks is permissible in patent applications, the proprietary nature of the marks should be respected and every effort made to prevent their use in any manner which might adversely affect their validity as trademarks.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the first paragraph of 35 U.S.C. 112:

   > The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

4. Claims 1 – 10 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention. The terms WINDOWS, DOS, LINUX, PENTIUM, VISUAL BASIC, and PERL are references to generic computer programs or hardware, and do not clearly identify the piece of software or hardware used.

Claim Rejections - 35 USC § 101

5. 35 U.S.C. 101 reads as follows:

   > Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

6. Claims 1 – 10 are rejected under 35 U.S.C. 101 because the claimed invention is directed to non-statutory subject matter. Claims 1 – 10 are for a "virtual machine system having a software processor", which is not a process, machine, manufacture, or composition of matter.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claim 1 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lee ("A Generic Virus Detection Agent on the Internet", 1997, Jieh-Sheng Lee, Jieh Hsiang, Po-Hao Tsang) in view of Le Charlier ("Dynamic Detection and Classification of Computer Viruses Using General Behavior Patterns", 1995, Baudouin Le Charlier, Morton Swimmer).

8.1. The art of Lee is directed to computer virus detection (page 210, document title) using a virtual system emulator by virtually executing computer code and analyzing the behavior.

8.2. Lee teaches a virtual machine system (page 214, section C. Software Emulator, lines 1 – 4) for computer code behavior analysis (page 214, section B. Components of VICE), the virtual machine system having a software processor (page 214, section C. Software Emulator, sentences 1 – 4).

8.3. Lee also teaches simulated memory (page 214, section C. Software Emulator, lines 7 – 9) and a simulated operating system representative of a host real computer system (page 214, section C. Software Emulator, lines 1 – 3, and lines 16 – 20, and page 217, section V. VICEd: Integrating VICE into the Internet,

paragraph 3), the computer code under analysis interacting with the simulated memory and the simulated operating system (page 214, section C. Software Emulator, lines 7 – 20) to generate the behavior flags (page 211, left-side column, paragraph 2).

8.3.1. Regarding page 211, left column, paragraph 2, a sequence of events is obviously a sequence of behavior flags.

8.4. Lee also teaches that the virtual machine passes data representative of the behavior record to the real host computer system prior to termination of the virtual machine (page 211, left side column, lines 32 – 36, and page 215, right side column, lines 14 – 16, and lines 23 - 26).

8.4.1. Regarding page 215, right side column, lines 14 – 16, the events from the emulation are obviously program behavior (page 211, left side column, lines 13 – 16 and lines 7 - 9).

8.4.2. Regarding page 211, left side column, lines 32 – 36, the virus analyzer is not the emulator, and therefore is part of the real host computer system.

8.5. Lee also teaches a behavior record storing behavior flags representative of computer code behavior observed by virtually executing the computer code under analysis within the virtual machine (page 211, column left-side, lines 13 – 16 and lines 7 - 9, and page 214, section C. Software Emulator, first paragraph).

8.5.1. Regarding page 211, left side column, lines 13 – 16 and lines 7 - 9, the phrase "summaries the execution as a sequence of events" is obviously producing a behavior record.

8.6. Lee does not specifically teach a sequencer that stores a sequence in which behavior flags are set in the behavior record during virtual execution of the computer code under analysis.

8.7. The art of Le Charlier is directed to computer virus detection (page 1, document title) using a virtual system emulator by virtually executing computer code and analyzing the behavior.

8.8. Le Charlier teaches a sequencer that stores a sequence in which behavior flags are set in the behavior record during virtual execution of the computer code under analysis (page 13, section 4.5, lines 1 – 4, and page 13, section 4.4).

8.8.1. Regarding page 13, section 4.5, lines 1 – 4, the time-stamp is obviously a data structure that tracks the evolution of the associated behavior.

8.8.2. Regarding page 13, section 4.4; this section documents the virtual execution of computer code under analysis.

8.9. Lee and Le Charlier are analogous art because they are both directed to a similar problem solving area, that of computer virus detection.

8.10. The motivation for combining the art of Le Charlier with the art of Lee would have been obvious in view of the analysis in Lee of the sequence in which program behaviors are performed (pages 216 – 216, section E. The Knowledge Base) which obviously requires a sequence of behavior flags.

8.11. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Le Charlier with the art of Lee for the benefit of obtaining the invention specified in claim 1.

9. Claim 2 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lee ("A Generic Virus Detection Agent on the Internet", 1997, Jieh-Sheng Lee, Jieh Hsiang, Po-Hao Tsang) in view of Le Charlier ("Dynamic Detection and Classification of Computer Viruses Using General Behavior Patterns", 1995, Baudouin Le Charlier, Morton Swimmer), further in view of Duncan ("Advanced MS-DOS", 1986, Ray Duncan, Published by Microsoft Press).

9.1. The art of Lee is directed to computer virus detection (page 210, title) using a virtual system emulator by virtually executing computer code and analyzing the behavior.

9.2. Lee teaches a virtual machine system (page 214, section C. Software Emulator, lines 1 – 4) for computer code behavior analysis (page 214, section B. Components of VICE), the virtual machine system having a software processor (page 214, section C. Software Emulator, sentences 1 – 4).

9.3. Lee also teaches a register or structure that stores behavior flags representative of computer code behavior observed by virtually executing the computer code under analysis within the virtual machine (page 211, left side column, lines 13 – 16 and lines 7 - 9, and page 214, section C. Software Emulator, first paragraph).

9.3.1. Regarding page 211, left side column, lines 13 – 16 and lines 7 - 9, the phrase "summaries the execution as a sequence of events" is obviously storing behavior flags in a register or structure.

9.4. Lee also teaches a memory structure simulating processor memory (page 214, section C. Software Emulator, lines 7 – 10).

9.5. Lee teaches a multi-path emulator is capable of emulating all executable paths in computer code under analysis within a virtual machine (page 215, left side column, lines 1 – 3, and page 214, section C.1 Multi-Path Emulation, paragraph 2). Lee also teaches in multi-path emulation to record all critical entry points during an execution (page 215, left side column, lines 6 – 7). Therefore, it would have been obvious to store all entry points to the computer code under analysis within the virtual machine

9.5.1. Regarding page 215, lines 2 – 3; it is obvious that the emulator is analyzing computer code (page 214, section C.1). A multi-path emulator is interpreted as a virtual machine.

9.5.2. Regarding page 215, left side column, lines 6 – 7; the applicant equates branches with entry points (specification, page 17, lines 10 – 14).

9.6. Lee also teaches one or more operating system simulation shells that simulate values returned by a real operating system under which the computer code under analysis is intended to operate (page 217, section V. VICEd: Integrating VICE into the Internet, lines 23 – 29, and page 214, section C. Software Emulator, lines 16 – 20, and page 219, section VI. Discussion and future work, paragraph 7).

9.6.1. Regarding page 217, section V. VICEd: Integrating VICE into the Internet, lines 23 – 29, since VICE is emulating DOS, the cited lines teach an operating system simulation shell.

9.6.2. Regarding page 214, section C. Software Emulator, lines 16 – 20, since interrupt service routines are part of the operating system, the cited lines teach to simulate values returned by a real operating system.

9.7. Lee does not specifically teach a structure that stores interrupt vector addresses, pointing at interrupt service routines loaded into memory reserved by the virtual machine when the virtual machine is initialized.

9.8. Lee does not specifically teach a register or structure that stores a sequence in which behavior flags are set in the behavior flags register or structure.

9.9. Lee does not specifically teach a memory structure simulating input and output ports.

9.10. Lee does not specifically teach an entry point table that stores all entry points to the computer code under analysis within the virtual machine.

9.11. The art of Le Charlier is directed to computer virus detection (page 1, title) using a virtual system emulator by virtually executing computer code and analyzing the behavior.

9.12. Le Charlier teaches a register or structure that stores a sequence in which behavior flags are set in the behavior flags register or structure (page 13, section 4.5, lines 1 – 4).

9.12.1. Regarding page 13, section 4.5, lines 1 – 4, the time-stamp is obviously a data structure that tracks the evolution of the associated behavior.

9.13. Le Charlier also teaches a memory structure simulating input and output ports (page 13, section 4.4, first paragraph).

9.13.1. Regarding page 13, section 4.4, first paragraph, it is obvious that the ports are simulated with memory structures containing software.

9.14. Lee and Le Charlier are analogous art because they are both directed to a similar problem solving area, that of computer virus detection.

9.15. The motivation for combining the art of Le Charlier with the art of Lee would have been obvious in view of the analysis in Lee of the sequence in which program behaviors are performed (pages 216 – 216, section E. The Knowledge Base) that obviously requires a register or structure that stores a sequence in which behavior flags are set in a behavior flags register or structure; and the need in Lee for a virtual system environment (page 214, section C., first paragraph) which obviously requires providing the features of a corresponding real system environment.

9.16. The art of Duncan is directed toward MS-DOS operating system.

9.17. Duncan teaches MS-DOS interrupt service routines (page 207, and page 208, first sentence) and interrupt vector addresses (page 208, top half of the page).

9.17.1. Regarding page 208, top half of the page; an interrupt vector is an interrupt vector address.

9.17.2. Regarding page 207, and page 208, first sentence, an interrupt handler is an interrupt service routine. An interrupt handler is included as a part of MS-DOS.

9.18. Duncan also teaches a structure that stores interrupt vector addresses (page 211, section The Interrupt Vector Table), pointing at interrupt service routines (page 211, section The Interrupt Vector Table).

9.18.1. Regarding page 211, section The Interrupt Vector Table, the interrupt vector table is a structure. An interrupt handler is an interrupt service routine. The interrupt vector table is included in MS-DOS.

9.19. Duncan also teaches that MS-DOS is loaded when the computer is initialized (pages 14 – 18, How MS-DOS Is Loaded). Therefore, as discussed above, it was common knowledge to the ordinary artisan at the time of invention that MS-DOS is loaded when the computer is initialized, and that MS-DOS includes a structure that stores interrupt vector addresses, pointing at interrupt service routines. It is obvious that when MS-DOS is loaded that it reserves memory for its structures. A virtual machine is interpreted as a virtual computer that simulates the operation of a real computer system. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention, that when building a virtual machine, to include a structure that stores interrupt vector addresses, pointing at interrupt service routines loaded into memory reserved by the virtual machine when the virtual machine is initialized.

9.20. Duncan also teaches an entry point table that stores entry point addresses (page 211, section The Interrupt Vector Table).

9.20.1. Regarding page 211, section The Interrupt Vector Table, the interrupt vector table is a table of entry point addresses into interrupt service routines.

9.21. Therefore, as discussed above, it was common knowledge to the ordinary artisan at the time of invention to use an entry point table that stores entry points.

9.22. Lee and Duncan are analogous art because they both have a similar problem solving area, that of building the structure and functions of an operating system.

9.23. The motivation for combining the art of Duncan with the art of Lee would have been obvious in view of the need in Lee to emulate DOS having virtual interrupt service routines (page 214, section C. Software Emulator, and page 217, section V. VICEd: Integrating VICE into the Internet, lines 23 – 29), and the need in Lee for a method to record all entry points for a multi-path emulator emulating all executable paths (page 215, left side column, second paragraph, and page 215, left side column, lines 2 – 3, and page 214, section C.1 Multi-Path Emulation, paragraph 2).

9.23.1. Regarding page 215, left side column, second paragraph, branching points are entry points (the applicant equates branches with entry points in the specification, page 17, lines 10 – 14).

9.23.2. Regarding page 217, section V. VICEd: Integrating VICE into the Internet, lines 23 – 29; MS-DOS is a version of DOS.

9.24. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan with the art of Lee to obtain a structure that stores interrupt vector addresses, pointing at interrupt service routines loaded into memory reserved by the virtual machine when the virtual machine is initialized; and to obtain an entry point table that stores all entry points to the computer code under analysis within the virtual machine.

9.25. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan and the art of Le Charlier with the art of Lee to obtain the invention specified in claim 2.

10. Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lee ("A Generic Virus Detection Agent on the Internet", 1997, Jieh-Sheng Lee, Jieh Hsiang, Po-Hao Tsang) in view of Le Charlier ("Dynamic Detection and Classification of Computer Viruses Using General Behavior Patterns", 1995, Baudouin Le Charlier, Morton Swimmer), further in view of Duncan ("Advanced MS-DOS", 1986, Ray Duncan, Published by Microsoft Press).

10.1. Claim 3 is a dependent claim of claim 2, and thereby incorporates all of the rejected limitations of claim 2.

10.2. The art of Lee is directed to computer virus detection (page 210, title) using a virtual system emulator by virtually executing computer code and analyzing the behavior.

10.3. Lee teaches a system wherein a software processor executes the computer code under analysis (page 214, section C. Software Emulator, lines 1 – 3), or fragments of the computer code under analysis (page 215, left side column, lines 26 – 39), and produces a behavior pattern comprising a set of behavior flags (page 211, left side column, lines 13 – 16 and lines 7 - 9).

10.3.1. Regarding page 215, left side column, lines 26 – 39; the one-path time-out would cause only a fragment of the computer code to execute.

10.3.2. Regarding page 211, left side column, lines 13 – 16 and lines 7 - 9; the phrase "summaries the execution as a sequence of events" is obviously producing a behavior pattern comprising a set of behavior flags.

10.4. Lee does not specifically teach a system wherein the software processor executes the computer code under analysis, or fragments of the computer code

under analysis, starting at each of the entry points defined within the entry point table and produces a behavior pattern comprising a set of behavior flags.

10.5. The art of Duncan is directed to the MS-DOS operating system.

10.6. Duncan teaches an entry point table that stores entry point addresses (page 211, section The Interrupt Vector Table).

10.6.1. Regarding page 211, section The Interrupt Vector Table, the interrupt vector table is a table of entry point addresses into interrupt service routines.

10.7. Therefore, as discussed above, it was common knowledge to the ordinary artisan at the time of invention to use an entry point table that stores entry point addresses.

10.8. Duncan also teaches executing computer code starting at each of the entry points defined within an entry point table (page 29, paragraph 2, first sentence, and page 30, figure 3-5, byte offset 0016H).

10.8.1. Regarding page 30, figure 3-5; the contents of the IP register at entry is an entry point address in an entry point table with a single entry.

10.8.2. Regarding page 29, paragraph 2, first sentence; the sentence teaches executing computer code starting at each of the entry points defined within an entry point table (see figure 3-5).

10.9. Lee and Duncan are analogous art because they both contain a similar problem solving area, that of building the structure and functions of an operating system.

10.10. The motivation for combining the art of Duncan with the art of Lee would have been obvious in view of the need in Lee for a method to record entry points (page 215, second paragraph), because Lee teaches a multi-path emulator is capable of emulating all executable program paths (page 215, lines 1 – 3, and

page 214, section C.1 Multi-Path Emulation, paragraph 2), where each branch point in the program is a possible entry point (page 215, lines 6 – 11).

10.10.1. Regarding page 215, second paragraph, branching points are entry points (the applicant equates branches with entry points in the specification, page 17, lines 10 – 14).

10.11. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan with the art of Lee to obtain a system wherein a software processor executes the computer code under analysis, or fragments of the computer code under analysis, starting at each of the entry points defined within the entry point table and produces a behavior pattern comprising a set of behavior flags.

10.12. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan and the art of Le Charlier with the art of Lee to obtain the invention specified in claim 3.

11. Claim 4 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lee ("A Generic Virus Detection Agent on the Internet", 1997, Jieh-Sheng Lee, Jieh Hsiang, Po-Hao Tsang) in view of Le Charlier ("Dynamic Detection and Classification of Computer Viruses Using General Behavior Patterns", 1995, Baudouin Le Charlier, Morton Swimmer), further in view of Duncan ("Advanced MS-DOS", 1986, Ray Duncan, Published by Microsoft Press), further in view of IBM Dictionary of Computing ("IBM Dictionary of Computing", George McDaniel, 1994, Tenth Edition, McGraw-Hill).

11.1. Claim 4 is a dependent claim of claim 2, and thereby incorporates ail of the rejected limitations of claim 2.

11.2. The art of Lee is directed to computer virus detection (page 210, title) using a virtual system emulator by virtually executing computer code and analyzing the behavior.

11.3. Lee teaches a system wherein the software processor executes the computer code under analysis (page 214, section C. Software Emulator, lines 1 – 3), and produces a sequence in which the behavior flags are set (page 211, left side column, lines 13 – 16 and lines 7 - 9).

11.3.1. Regarding page 211, left side column, lines 13 – 16 and lines 7 - 9; the phrase "summaries the execution as a sequence of events" is obviously producing a sequence in which behavior flags are set.

11.4. Lee does not specifically teach a system wherein the software processor executes the computer code under analysis, starting at each entry point defined within the entry point table and produces a sequence in which the behavior flags are set or reset.

11.5. The art of Duncan is directed to the MS-DOS operating system.

11.6. Duncan teaches an entry point table that stores entry point addresses (page 211, section The Interrupt Vector Table).

11.6.1. Regarding page 211, section The Interrupt Vector Table, the interrupt vector table is a table of entry point addresses into interrupt service routines.

11.7. Therefore, as discussed above, it was common knowledge to the ordinary artisan at the time of invention to use an entry point table that stores entry point addresses.

11.8. Duncan also teaches executing computer code starting at each entry point defined within an entry point table (page 29, paragraph 2, first sentence, and page 30, figure 3-5, byte offset 0016H).

11.8.1. Regarding page 30, figure 3-5; the contents of the IP register at entry is an entry point address in an entry point table with a single entry.

11.8.2. Regarding page 29, paragraph 2, first sentence; the sentence teaches executing computer code starting at each entry point defined within an entry point table (see figure 3-5).

11.9. Lee and Duncan are analogous art because they both contain a similar problem solving area, that of building the structure and functions of an operating system.

11.10. The motivation for combining the art of Duncan with the art of Lee would have been obvious in view of the need in Lee for a method to record entry points (page 215, second paragraph), because Lee teaches a multi-path emulator is capable of emulating all executable program paths (page 215, lines 1 – 3, and page 214, section C.1 Multi-Path Emulation, paragraph 2), where each branch point in the program is a possible entry point (page 215, lines 6 -- 11).

11.10.1. Regarding page 215, second paragraph, branching points are entry points (the applicant equates branches with entry points in the specification, page 17, lines 10 – 14).

11.11. The art of IBM Dictionary of Computing is directed to common knowledge of an artisan of ordinary skill in the art of computer software development.

11.12. IBM Dictionary of Computing teaches to reset a behavior flag (page 576, entry (2) for reset).

11.13. Therefore, as discussed above, it was common knowledge to an artisan of ordinary skill in the art at the time of invention to reset behavior flags.

11.14. Lee and IBM Dictionary of Computing are analogous art because they both include components directed to a similar problem solving area, the problem of computer software development.

11.15. The motivation for combining the art of IBM Dictionary of Computing with the art of Lee would have been obvious in view of the teaching in Lee for a multi-path emulator (page 214, left side column, lines 2 – 3) which would obviously need to reset behavior flags for each emulated path.

11.16. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan and the art of IBM Dictionary of Computing with the art of Lee to obtain a system wherein a software processor executes the computer code under analysis, starting at each entry point defined within the entry point table and produces a sequence in which the behavior flags are set or reset.

11.17. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan and the art of IBM Dictionary of Computing with the art of Lee to obtain the invention specified in claim 4.

12. Claim 5 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lee ("A Generic Virus Detection Agent on the Internet", 1997, Jieh-Sheng Lee, Jieh Hsiang, Po-Hao Tsang) in view of Le Charlier ("Dynamic Detection and Classification of Computer Viruses Using General Behavior Patterns", 1995, Baudouin Le Charlier, Morton Swimmer), further in view of Duncan ("Advanced MS-DOS", 1986, Ray Duncan, Published by Microsoft Press).

12.1. Claim 5 is a dependent claim of claim 2, and thereby incorporates all of the rejected limitations of claim 2.

12.2. Lee teaches a system wherein a software processor interprets a high level language within the virtual machine (page 219, section VI. Discussion and future work, paragraph 5, lines 2 – 3, and page 219, section VI. Discussion and future work, paragraph 6).

12.2.1. Regarding page 219, section VI. Discussion and future work, paragraph 5, lines 2 – 3, and page 219, section VI. Discussion and future work, paragraph 6; Java was a high level language, and the Java virtual machine utilized a software processor that interprets Java (Page 41, subsection labeled "Run options for Java applications", in "Programmer's Guide PowerJ", Sybase, 1996-1997).

13. Claim 6 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lee ("A Generic Virus Detection Agent on the Internet", 1997, Jieh-Sheng Lee, Jieh Hsiang, Po-Hao Tsang) in view of Le Charlier ("Dynamic Detection and Classification of Computer Viruses Using General Behavior Patterns", 1995, Baudouin Le Charlier, Morton Swimmer), further in view of Duncan ("Advanced MS-DOS", 1986, Ray Duncan, Published by Microsoft Press).

13.1. Claim 6 is a dependent claim of claim 5, and thereby incorporates all of the rejected limitations of claim 5.

13.2. The art of Lee is directed to computer virus detection (page 210, title) using a virtual system emulator by virtually executing computer code and analyzing the behavior.

13.3. Lee teaches a system wherein a software processor executes the computer code under analysis (page 214, section C. Software Emulator, lines 1 – 3), or fragments of the computer code under analysis (page 215, left side column, lines 26 – 39), and produces a behavior pattern comprising a set of behavior flags (page 211, left side column, lines 13 – 16 and lines 7 – 9).

13.3.1. Regarding page 215, left side column, lines 26 – 39; the one-path time-out would cause only a fragment of the computer code to execute.

13.3.2. Regarding page 211, left side column, lines 13 – 16 and lines 7 - 9; the phrase "summaries the execution as a sequence of events" is obviously producing a behavior pattern comprising a set of behavior flags.

13.4. Lee does not specifically teach a system wherein the software processor executes the computer code under analysis, starting at each of the entry points defined within the entry point table and produces a behavior pattern comprising a set of behavior flags.

13.5. The art of Duncan is directed to the MS-DOS operating system.

13.6. Duncan teaches an entry point table that stores entry point addresses (page 211, section The Interrupt Vector Table).

13.6.1. Regarding page 211, section The Interrupt Vector Table, the interrupt vector table is a table of entry point addresses into interrupt service routines.

13.7. Therefore, as discussed above, it was common knowledge to the ordinary artisan at the time of invention to use an entry point table that stores entry point addresses.

13.8. Duncan also teaches executing computer code starting at each of the entry points defined within an entry point table (page 29, paragraph 2, first sentence, and page 30, figure 3-5, byte offset 0016H).

13.8.1. Regarding page 30, figure 3-5; the contents of the IP register at entry is an entry point address in an entry point table with a single entry.

13.8.2. Regarding page 29, paragraph 2, first sentence; the sentence teaches executing computer code starting at each of the entry points defined within an entry point table (see figure 3-5).

13.9. Lee and Duncan are analogous art because they both contain a similar problem solving area, that of building the structure and functions of an operating system.

13.10. The motivation for combining the art of Duncan with the art of Lee would have been obvious in view of the need in Lee for a method to record entry points (page 215, second paragraph), because Lee teaches a multi-path emulator is capable of emulating all executable program paths (page 215, lines 1 – 3, and page 214, section C.1 Multi-Path Emulation, paragraph 2), where each branch point in the program is a possible entry point (page 215, lines 6 – 11).

13.10.1. Regarding page 215, second paragraph, branching points are entry points (the applicant equates branches with entry points in the specification, page 17, lines 10 – 14).

13.11. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan with the art of Lee to obtain a system wherein a software processor executes the computer code under analysis, or fragments of the computer code under analysis, starting at each of the entry points defined within the entry point table and produces a behavior pattern comprising a set of behavior flags.

13.12. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan and the art of Le Charlier with the art of Lee to obtain the invention specified in claim 6.

14. Claim 7 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lee ("A Generic Virus Detection Agent on the Internet", 1997, Jieh-Sheng Lee, Jieh Hsiang, Po-Hao Tsang) in view of Le Charlier ("Dynamic Detection and Classification of Computer Viruses Using General Behavior Patterns", 1995, Baudouin Le Charlier, Morton Swimmer), further in view of Duncan ("Advanced MS-DOS", 1986, Ray Duncan, Published by Microsoft Press), further in view of IBM Dictionary of Computing ("IBM Dictionary of Computing", George McDaniel, 1994, Tenth Edition, McGraw-Hill).

14.1. Claim 7 is a dependent claim of claim 5, and thereby incorporates all of the rejected limitations of claim 5.

14.2. The art of Lee is directed to computer virus detection (page 210, title) using a virtual system emulator by virtually executing computer code and analyzing the behavior.

14.3. Lee teaches a system wherein the software processor executes the computer code under analysis (page 214, section C. Software Emulator, lines 1 – 3), and produces a sequence in which the behavior flags are set (page 211, left side column, lines 13 – 16 and lines 7 - 9).

14.3.1. Regarding page 211, left side column, lines 13 – 16 and lines 7 - 9; the phrase "summaries the execution as a sequence of events" is obviously producing a sequence in which behavior flags are set.

14.4. Lee does not specifically teach a system wherein the software processor executes the computer code under analysis, starting at each entry point defined within the entry point table and produces a sequence in which the behavior flags are set or reset.

14.5. The art of Duncan is directed to the MS-DOS operating system.

14.6. Duncan teaches an entry point table that stores entry point addresses (page 211, section The Interrupt Vector Table).

14.6.1. Regarding page 211, section The Interrupt Vector Table, the interrupt vector table is a table of entry point addresses into interrupt service routines.

14.7. Therefore, as discussed above, it was common knowledge to the ordinary artisan at the time of invention to use an entry point table that stores entry point addresses.

14.8. Duncan also teaches executing computer code starting at each entry point defined within an entry point table (page 29, paragraph 2, first sentence, and page 30, figure 3-5, byte offset 0016H).

14.8.1. Regarding page 30, figure 3-5; the contents of the IP register at entry is an entry point address in an entry point table with a single entry.

14.8.2. Regarding page 29, paragraph 2, first sentence; the sentence teaches executing computer code starting at each entry point defined within an entry point table (see figure 3-5).

14.9. Lee and Duncan are analogous art because they both contain a similar problem solving area, that of building the structure and functions of an operating system.

14.10. The motivation for combining the art of Duncan with the art of Lee would have been obvious in view of the need in Lee for a method to record entry points (page 215, second paragraph), because Lee teaches a multi-path emulator is capable of emulating all executable program paths (page 215, lines 1 – 3, and page 214, section C.1 Multi-Path Emulation, paragraph 2), where each branch point in the program is a possible entry point (page 215, lines 6 – 11).

14.10.1. Regarding page 215, second paragraph, branching points are entry points (the applicant equates branches with entry points in the specification, page 17, lines 10 – 14).

14.11. The art of IBM Dictionary of Computing is directed to common knowledge of an artisan of ordinary skill in the art of computer software development.

14.12. IBM Dictionary of Computing teaches to reset a behavior flag (page 576, entry (2) for reset).

14.13. Therefore, as discussed above, it was common knowledge to an artisan of ordinary skill in the art at the time of invention to reset behavior flags.

14.14. Lee and IBM Dictionary of Computing are analogous art because they both include components directed to a similar problem solving area, the problem of computer software development.

14.15. The motivation for combining the art of IBM Dictionary of Computing with the art of Lee would have been obvious in view of the teaching in Lee for a multi-path emulator (page 214, left side column, lines 2 – 3) which would obviously need to reset behavior flags for each emulated path.

14.16. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan and the art of IBM Dictionary of Computing with the art of Lee to obtain a system wherein a software processor executes the computer code under analysis, starting at each entry point defined within the entry point table and produces a sequence in which the behavior flags are set or reset.

14.17. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan and the art of IBM Dictionary of Computing with the art of Lee to obtain the invention specified in claim 7.

15. Claim 8 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lee ("A Generic Virus Detection Agent on the Internet", 1997, Jieh-Sheng Lee, Jieh Hsiang, Po-Hao Tsang) in view of Le Charlier ("Dynamic Detection and Classification of Computer Viruses Using General Behavior Patterns", 1995, Baudouin Le Charlier, Morton Swimmer), further in view of Duncan ("Advanced MS-DOS", 1986, Ray Duncan, Published by Microsoft Press), further in view of Burd ("Systems Architecture", Stephen B. Burd, 1998, Second Edition, Course Technology), further in view of Custer("Inside Windows NT", Helen Custer, 1993, Microsoft Press).

15.1. Claim 8 is a dependent claim of claim 2, and thereby incorporates all of the rejected limitations of claim 2.

15.2. The art of Lee is directed to computer virus detection (page 210, title) using a virtual system emulator by virtually executing computer code and analyzing the behavior.

15.3. Lee does not teach a system wherein the software processor executes 32-bit or 64-bit program code and the operating system simulation shell responds to application program interface calls.

15.4. The art of Custer is directed to the structure and functions of an operating system, including a virtual system emulator (pages 149 – 152).

15.5. Custer teaches a system wherein the software processor executes 32-bit program code (page 151, second paragraph, third sentence to the end of the paragraph) and the operating system simulation shell responds to application program interface calls (page 150, second paragraph [line 10 onward], and page 151, paragraphs 2 and 3).

15.5.1. Regarding page 150, second paragraph [line 10 onward], and page 151, paragraphs 2 and 3; since the MS-DOS operating system is being simulated, it is obvious that the operating system simulation shell responds to application program interface calls.

15.6. Therefore, as discussed above, it was common knowledge to an artisan of ordinary skill at the time of invention to have a system wherein a software processor executes 32-bit program code and the operating system simulation shell responds to application program interface calls.

15.7. The art of Burd is directed to computer systems architecture, including functions of operating systems (chapter 12 and chapter 14), and processor technology and architecture (chapter 5).

15.8. Burd teaches a 32-bit processor and a 64-bit processor (page 171, lines 18 – 19, and pages 171 – 172)

15.9. Therefore, as discussed above, 32-bit and 64-bit processors were common knowledge to an artisan of ordinary skill at the time of invention.

15.10. Lee and Custer are analogous art because they both include components directed to a similar problem solving area, that of building the structure and functions of an operating system.

15.11. The motivation for combining the art of Custer with the art of Lee would have been obvious in view of the need in Lee for an operating system emulator, and the solution provided in Custer for an operating system emulator.

15.12. Lee and Burd are analogous art because they both include components directed to a similar problem solving area, that of building the structure and functions of an operating system.

15.13. The motivation for combining the art of Burd with the art of Lee would have been obvious in view of the expectation of the benefit of the capability to perform simulation of an expanded base of processors, both 32-bit and 64-bit.

15.14. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Custer and the art of Burd with the art of Lee to obtain the invention specified in claim 8.

16. Claim 9 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lee ("A Generic Virus Detection Agent on the Internet", 1997, Jieh-Sheng Lee, Jieh Hsiang, Po-Hao Tsang) in view of Le Charlier ("Dynamic Detection and Classification of Computer Viruses Using General Behavior Patterns", 1995, Baudouin Le Charlier, Morton Swimmer), further in view of Duncan ("Advanced MS-DOS", 1986, Ray Duncan, Published by Microsoft Press), further in view of Burd ("Systems Architecture", Stephen B. Burd, 1998, Second Edition, Course Technology), further in view of Custer("Inside Windows NT", Helen Custer, 1993, Microsoft Press).

16.1. Claim 9 is a dependent claim of claim 8, and thereby incorporates all of the rejected limitations of claim 8.

16.2. The art of Lee is directed to computer virus detection (page 210, title) using a virtual system emulator by virtually executing computer code and analyzing the behavior.

16.3. Lee teaches a system wherein a software processor executes the computer code under analysis (page 214, section C. Software Emulator, lines 1 – 3), or fragments of the computer code under analysis (page 215, left side column, lines 26 – 39), and produces a behavior pattern comprising a set of behavior flags (page 211, left side column, lines 13 – 16 and lines 7 - 9).

16.3.1. Regarding page 215, left side column, lines 26 – 39; the one-path time-out would cause only a fragment of the computer code to execute.

16.3.2. Regarding page 211, left side column, lines 13 – 16 and lines 7 - 9; the phrase "summaries the execution as a sequence of events" is obviously producing a behavior pattern comprising a set of behavior flags.

16.4. Lee does not specifically teach a system wherein the software processor executes the computer code under analysis, or fragments of the computer code under analysis, starting at each of the entry points defined within the entry point table and produces a behavior pattern comprising a set of behavior flags.

16.5. The art of Duncan is directed to the MS-DOS operating system.

16.6. Duncan teaches an entry point table that stores entry point addresses (page 211, section The Interrupt Vector Table).

16.6.1. Regarding page 211, section The Interrupt Vector Table, the interrupt vector table is a table of entry point addresses into interrupt service routines.

16.7. Therefore, as discussed above, it was common knowledge to the ordinary artisan at the time of invention to use an entry point table that stores entry point addresses.

16.8. Duncan also teaches executing computer code starting at each of the entry points defined within an entry point table (page 29, paragraph 2, first sentence, and page 30, figure 3-5, byte offset 0016H).

16.8.1. Regarding page 30, figure 3-5; the contents of the IP register at entry is an entry point address in an entry point table with a single entry.

16.8.2. Regarding page 29, paragraph 2, first sentence; the sentence teaches executing computer code starting at each of the entry points defined within an entry point table (see figure 3-5).

16.9. Lee and Duncan are analogous art because they both contain a similar problem solving area, that of building the structure and functions of an operating system.

16.10. The motivation for combining the art of Duncan with the art of Lee would have been obvious in view of the need in Lee for a method to record entry points (page 215, second paragraph), because Lee teaches a multi-path emulator is capable of emulating all executable program paths (page 215, lines 1 – 3, and page 214, section C.1 Multi-Path Emulation, paragraph 2), where each branch point in the program is a possible entry point (page 215, lines 6 – 11).

16.10.1. Regarding page 215, second paragraph, branching points are entry points (the applicant equates branches with entry points in the specification, page 17, lines 10 – 14).

16.11. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan with the art of Lee to obtain a system wherein a software processor executes the computer code under analysis, or fragments of the computer code under analysis, starting

at each of the entry points defined within the entry point table and produces a behavior pattern comprising a set of behavior flags.

16.12. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan and the art of Le Charlier with the art of Lee to obtain the invention specified in claim 8.

17. Claim 10 is rejected under 35 U.S.C. 103(a) as being unpatentable over Lee ("A Generic Virus Detection Agent on the Internet", 1997, Jieh-Sheng Lee, Jieh Hsiang, Po-Hao Tsang) in view of Le Charlier ("Dynamic Detection and Classification of Computer Viruses Using General Behavior Patterns", 1995, Baudouin Le Charlier, Morton Swimmer), further in view of Duncan ("Advanced MS-DOS", 1986, Ray Duncan, Published by Microsoft Press), further in view of IBM Dictionary of Computing ("IBM Dictionary of Computing", George McDaniel, 1994, Tenth Edition, McGraw-Hill).

17.1. Claim 10 is a dependent claim of claim 8, and thereby incorporates all of the rejected limitations of claim 8.

17.2. The art of Lee is directed to computer virus detection (page 210, title) using a virtual system emulator by virtually executing computer code and analyzing the behavior.

17.3. Lee teaches a system wherein the software processor executes the computer code under analysis (page 214, section C. Software Emulator, lines 1 – 3), and produces a sequence in which the behavior flags are set (page 211, left side column, lines 13 – 16 and lines 7 - 9).

17.3.1. Regarding page 211, left side column, lines 13 – 16 and lines 7 - 9; the phrase "summaries the execution as a sequence of events" is obviously producing a sequence in which behavior flags are set.

17.4. Lee does not specifically teach a system wherein the software processor executes the computer code under analysis, starting at each entry point defined within the entry point table and produces a sequence in which the behavior flags are set or reset.

17.5. The art of Duncan is directed to the MS-DOS operating system.

17.6. Duncan teaches an entry point table that stores entry point addresses (page 211, section The Interrupt Vector Table).

17.6.1. Regarding page 211, section The Interrupt Vector Table, the interrupt vector table is a table of entry point addresses into interrupt service routines.

17.7. Therefore, as discussed above, it was common knowledge to the ordinary artisan at the time of invention to use an entry point table that stores entry point addresses.

17.8. Duncan also teaches executing computer code starting at each entry point defined within an entry point table (page 29, paragraph 2, first sentence, and page 30, figure 3-5, byte offset 0016H).

17.8.1. Regarding page 30, figure 3-5; the contents of the IP register at entry is an entry point address in an entry point table with a single entry.

17.8.2. Regarding page 29, paragraph 2, first sentence; the sentence teaches executing computer code starting at each entry point defined within an entry point table (see figure 3-5).

17.9. Lee and Duncan are analogous art because they both contain a similar problem solving area, that of building the structure and functions of an operating system.

17.10. The motivation for combining the art of Duncan with the art of Lee would have been obvious in view of the need in Lee for a method to record entry points (page 215, second paragraph), because Lee teaches a multi-path emulator is

capable of emulating all executable program paths (page 215, lines 1 – 3, and page 214, section C.1 Multi-Path Emulation, paragraph 2), where each branch point in the program is a possible entry point (page 215, lines 6 – 11).

17.10.1. Regarding page 215, second paragraph, branching points are entry points (the applicant equates branches with entry points in the specification, page 17, lines 10 – 14).

17.11. The art of IBM Dictionary of Computing is directed to common knowledge of an artisan of ordinary skill in the art of computer software development.

17.12. IBM Dictionary of Computing teaches to reset a behavior flag (page 576, entry (2) for reset).

17.13. Therefore, as discussed above, it was common knowledge to an artisan of ordinary skill in the art at the time of invention to reset behavior flags.

17.14. Lee and IBM Dictionary of Computing are analogous art because they both include components directed to a similar problem solving area, the problem of computer software development.

17.15. The motivation for combining the art of IBM Dictionary of Computing with the art of Lee would have been obvious in view of the teaching in Lee for a multi-path emulator (page 214, left side column, lines 2 – 3) which would obviously need to reset behavior flags for each emulated path.

17.16. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan and the art of IBM Dictionary of Computing with the art of Lee to obtain a system wherein a software processor executes the computer code under analysis, starting at each entry point defined within the entry point table and produces a sequence in which the behavior flags are set or reset.

17.17. Therefore, as discussed above, it would have been obvious to the ordinary artisan at the time of invention to use the art of Duncan and the art of

IBM Dictionary of Computing with the art of Lee to obtain the invention specified in claim 10.

Conclusion

18. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

18.1. Nachenberg, Carey, "Computer Virus-Coevolution", 1997, Communications of the ACM, Vol. 40, No. 1

18.2. "Understanding Heuristics: Symantec's Bloodhound Technology", 1997, Symantec White Paper Series, Volume XXXIV

19. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Russell L. Guill whose telephone number is 571-272-7955. The examiner can normally be reached on 7:30 AM - 4 PM Monday - Friday.

20. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kevin Teska can be reached on 571-272-3716. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

21. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

RLG

KEVIN J. TESKA
SUPERVISORY
PATENT EXAMINER